

IS 6733 – Homework Submission Instructions

As stated in the syllabus, you need to submit both a PDF file of your Colab notebook codes and outputs and a shared link to your colab code for the homework assignments.

1. PDF file from your Colab code and output

Go the Colab "File" tab, select "Print", then in the destination, choose "Save as PDF". There are multiple configurations you can set to make your file to better display as shown here:

The image shows a Colab notebook interface. On the left, there is a code cell with Python code for sequence processing using Keras and TensorFlow. The code includes comments and prints the shapes of the training and testing data. On the right, the 'Print' menu is open, showing options for destination, pages, layout, and more settings. Red circles and arrows highlight specific settings: 'Save as PDF' in the destination dropdown, 'Landscape' in the layout dropdown, and 'Custom' in the scale dropdown. A red arrow points from the 'Scale' dropdown to the text 'You can set a smaller value for the scale to shrink the font size to better fit the page'.

```
import tensorflow as tf
from tensorflow import keras

- Sequence processing with convnets

This notebook contains the code samples found in Chapter 6, Section 4 of Deep Learning with Python. Note that the original text features far more content, in particular further explanations and figures: in this notebook, you will only find source code and related comments.

Implementing a 1D convnet

In Keras, you would use a 1D convnet via the Conv1D layer, which has a very similar interface to Conv2D. It takes as input 3D tensors with shape (samples, time, features) and also returns similarly-shaped 3D tensors. The convolution window is a 1D window on the temporal axis, axis 1 in the input tensor.

Let's build a simple 2-layer 1D convnet and apply it to the IMDB sentiment classification task that you are already familiar with.

As a reminder, this is the code for obtaining and preprocessing the data:

from keras.datasets import imdb
from keras.preprocessing import sequence

max_features = 10000 # number of words to consider as features
max_len = 500 # cut texts after this number of words (among top max_features most common words)

print('Loading data...')
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)
print(len(x_train), 'train sequences')
print(len(x_test), 'test sequences')

print('Pad sequences (samples x time)')
x_train = sequence.pad_sequences(x_train, maxlen=max_len)
x_test = sequence.pad_sequences(x_test, maxlen=max_len)
print('x_train shape:', x_train.shape)
print('x_test shape:', x_test.shape)

D. Using TensorFlow backend.
Loading data...
Downloading data from https://s3.amazonaws.com/text-datasets/imdb_pos
1745347/144789 [=====] - 50 Sub/step
25000 train sequences
25000 test sequences
Pad sequences (samples x time)
x_train shape: (25000, 500)
x_test shape: (25000, 500)

1D convnets are structured in the same way as their 2D counterparts that you have used in Chapter 5: they consist of a stack of Conv1D and MaxPooling1D layers, eventually ending in either a global pooling layer or a Flatten layer, turning the 3D outputs into 2D outputs, allowing to add one or more Dense layers to the model, for classification or regression.

One difference, though, is the fact that we can afford to use larger convolution windows with 1D convnets. Indeed, with a 2D convolution layer, a 3x3 convolution window contains 3*3 = 9 feature vectors, but with a 1D convolution layer, a convolution window of size 3 would only contain 3 feature vectors. We can thus easily afford 1D convolution windows of size 7 or 9.

This is our example 1D convnet for the IMDB dataset:

from keras.models import Sequential
from keras import layers
from keras.optimizers import RMSprop

model = Sequential()
model.add(layers.Embedding(max_features, 128, input_length=max_len))
model.add(layers.Conv1D(32, 7, activation='relu'))
model.add(layers.MaxPooling1D(3))
model.add(layers.Conv1D(32, 7, activation='relu'))
model.add(layers.GlobalMaxPooling1D())
model.add(layers.Dense(1))
```

Print 9 pages

Destination **Save as PDF**

Pages **All**

Layout **Portrait** ☒ **Landscape**

More settings

Paper size **Letter**

Pages per sheet **1**

Margins **Default**

Scale **Custom** 72

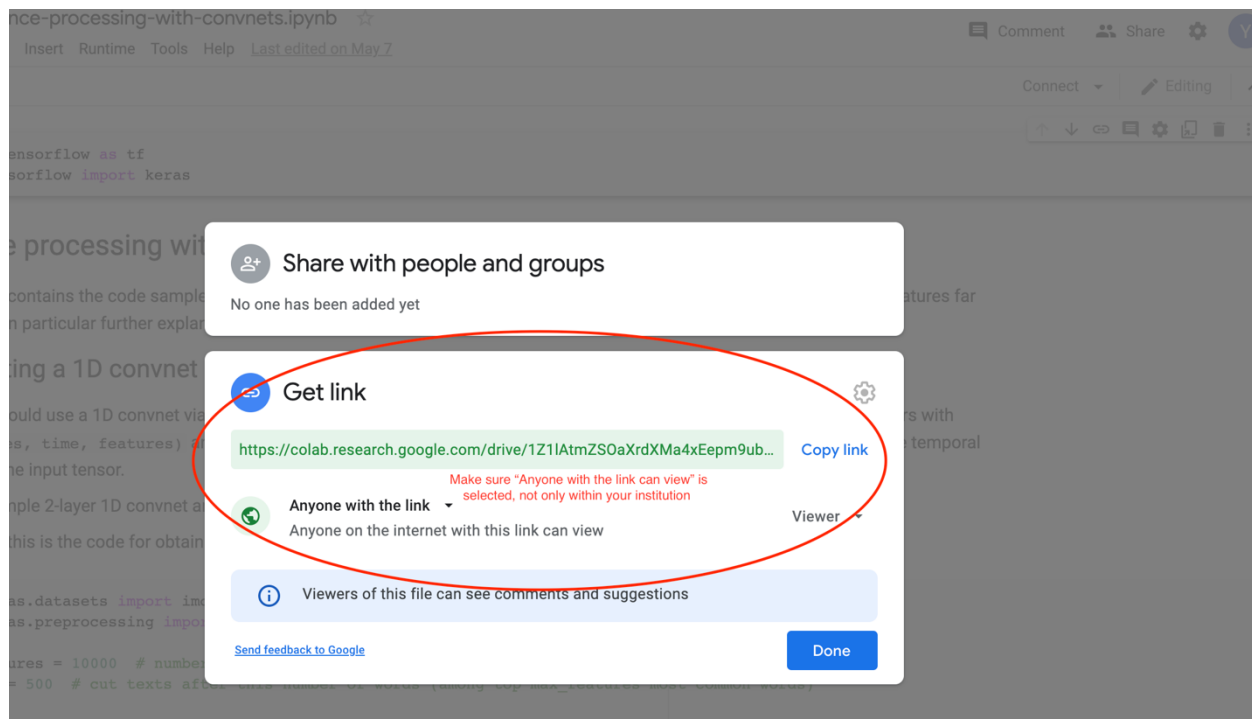
Options ☐ Headers and footers

Cancel Save

Make sure all your major code lines are displayed correctly by choosing the "Scale" and "Layout". If you cannot find the "Save as PDF" option in your browser, you may want to install the free Adobe Reader software.

2. Shared link to your Colab code and output

To get a sharable link from Colab, you need to first find the "Share" option on the top right of the colab page, and then get the shared link as shown below:



Do not directly copy the URL link from your web browser as that will not work. Moreover, you need to change the access right to "Anyone with the link can view", not "Anyone at the UTSA with the link can view" (This is a common mistake if you create the Gmail account using your UTSA email).

3. GPU use in Colab

When introducing Colab before, we mentioned that its key advantage is the free GPU. Using GPU is necessary for most deep learning codes as the computation workload is very high. Therefore, when running the deep learning codes given in the lectures as well as doing your homework, you will want to use the GPU to speed up the training process.

The instruction for using GPU in Colab is re-attached here for your information. (runtime --> change runtime type --> GPU). Keep in mind that this is a free and shared service, meaning that sometimes it is very fast and other times it could be slow.

CO IS6733-L2b-Pandas.ipynb ☆

File Edit View Insert Runtime Tools Help Last edited on October 21

+ Code + Text

!pip install pa

Requirement alr
Requirement alr
Requirement alr
Requirement alr
Requirement alr

[] import pandas a

▼ Creating Series

A Pandas Series is a one-dimensional array. It can be created from a list or array as

[] data = pd.Series([0.25, 0.5, 0.75, 1.0])
data

0 0.25

Runtime menu options:

- Run all ⌘/Ctrl+F9
- Run before ⌘/Ctrl+F8
- Run the focused cell ⌘/Ctrl+Enter
- Run selection ⌘/Ctrl+Shift+Enter
- Run after ⌘/Ctrl+F10
- Interrupt execution ⌘/Ctrl+M
- Restart runtime ⌘/Ctrl+M
- Restart and run all
- Factory reset runtime
- Change runtime type
- Manage sessions
- View runtime logs

created from a list or array as follows:

Notebook settings

Hardware accelerator

GPU ?

To get the most out of Colab, avoid using a GPU unless you need one. [Learn more](#)

☐ Omit code cell output when saving this notebook

CANCEL

SAVE

Here are some additional good practices:

1. In Colab, please use a separate code cell for addressing each subquestion and displaying the output result. You can observe the step-by-step result as you analyze the data and correct some potential mistakes early. This feature is actually the key advantage of Colab/Jupyter Notebook compared to other Python editors (e.g., VS Code and PyCharm). Also, in terms of generating PDF file in Colab, this will make your PDF file much clearer.

2. Make sure your code is executable on Colab. Some of you write the code locally, but after uploading it to Colab, it might not work due to some file path issues. The best way is to write the code directly on Colab and use URL to fetch the data instead of local file path. Furthermore, you will see in the future lectures that when we run some deep learning codes, the free GPU on Colab will greatly speed up the training process compared to your laptop. So you will want to use Colab if you do not have a powerful deep learning workstation.